Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 52 (2015) 980 – 987

Procedia Computer Science

2nd International Workshop on Computational Antifragility and Antifragile Engineering (ANTIFRAGILE 2015)

Automatic Resource Allocation for High Availability Cloud Services

Stefano Marrone[a,*], Roberto Nardone[b]

[a]*Dipartimento di Matematica e Fisica, Seconda Universitá di Napoli*

[b]*Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione, Universitá "Federico II" di Napoli*

Abstract

This paper proposes an approach to support cloud brokers finding optimal configurations in the deployment of dependability and security sensitive cloud applications. The approach is based on model-driven principles and uses both UML and Bayesian Networks to capture, analyse and optimise cloud deployment configurations. While the paper is most focused on the initial allocation phase, the approach is extensible to the operational phases of the life-cycle. In such a way, a continuous improvement of cloud applications may be realised by monitoring, enforcing and re-negotiating cloud resources following detected anomalies and failures.

Peer-review under responsibility of the Conference Program Chairs

Keywords: Virtual Machine Allocation; Bayesian Networks; Resiliency; Model-Driven Engineering; MARTE-DAM

1. Introduction

Entrepreneurs are more and more willing to implement highly available and scalable web applications to improve the volume of their business: cloud computing paradigm is one of the keys to pursue this goal. One of the most important tools to develop and distribute (cloud-based) services is the Service Level Agreement (SLA) mechanism; by means of an SLA, an End-User (EU) and a Cloud Service Provider (CSP) can agree on the service to provide and on the Quality of Service (QoS) requirements the CSP must fulfil. Availability is one of the most sensitive features of cloud computing as reported in some white papers and research surveys [1,2]. Notwithstanding the problems to pass from an availability-oriented specification to an effective deployment of virtual resources has been studied, the integration of the different phases of cloud service life-cycle during also run-time is still an open research topic.

This paper aims at helping the building of highly available cloud services by means of model-driven engineering. This technique has been described having not fully realised its great potential [3]: however, it still remains a valid mean specifically when used to generate formal models. The paper provides an automatable methodology whose first step requires the definition of a high level description of cloud services, underlying software components, as well as the description of user level availability requirements. Starting from this model, a Bayesian Network (BN) model is generated and then it is used as input to an optimisation activity. This optimisation modifies both the structure and the

* Viale Lincoln, 5. 81100 - Caserta, Italy. Tel.: +39 0823 27 5101; fax: +39 0823 27 4753.
E-mail address: stefano.marrone@unina2.it

Peer-review under responsibility of the Conference Program Chairs
doi:10.1016/j.procs.2015.05.176

parameters of the BN model to find the most fitting allocation of services on virtual and physical resources provided by external CSPs. In particular, the paper focuses on both high level and formal modelling; it also gives some hints about the transformation and the optimisation concerns. Once the (sub-)optimal model is computed, it can be used to generate SLAs able to capture the different Virtual Machines (VMs) configuration with their specific availability requirements. This work constitutes the first step of a wider research whose aim is to define an autonomic system able to: *plan* resources able to fulfil functional and availability related EU requirements; *monitor* instantiated resources by measuring availability metrics; *run* effective counteractions if one or more Service Level Objectives (SLOs), the single elements of a signed SLA, is going to be or is violated.

The context of this work is SPECS, an ongoing EU research project of the FP7-ICT programme[4]. The aim of SPECS is to respond to the cloud community need of improving comprehensibly the state-of-the-art in cloud computing security by creating, promoting and exploiting a user-centric framework and a platform dedicated to offer Security-as-a-Service using an SLA-based approach. This work focuses on availability, one of the security-oriented features; this notwithstanding, the proposed approach is general enough to be extended in the future to other security concerns. In the SPECS project a typical SLA life cycle can be characterised by three main phases that are cyclically conducted: Negotiation, Monitoring and Enforcement. In the *Negotiation* phase, the SLA is not fully defined, and customer(s) and provider(s) conduct a negotiation process on requirements/services to find agreement on what the SLA should effectively offer. During the *Monitoring* phase, a signed SLA is checked for its actual degree of conformance or for penalties if violated. Monitoring implies (a) verifying that SLAs are respected, and (b) generating alerts before SLAs are not fulfilled. The final step of the SLA life cycle is the *Enforcement*, where the actions needed to respect the SLA (i.e., to keep a sustained level of security) are effectively taken. The three phases are correlated: the Negotiation cannot be performed without taking in consideration how SLA can be granted, i.e., how the Enforcement will take place. Enforcement needs Monitoring in order to evaluate the real state of the solution before applying the predefined policies and procedures, while Monitoring needs the results of negotiation to know what to monitor and which alerts should be generated.

The rest of the paper is structured as follows. Section 2 provides both an overview of the related research as well as briefly introduction to background information. Section 3 depicts the approach at-a-glance and introduces a running example; such example is used to show the details of the different models in Section 4. Section 5 ends the paper showing future research developments. The reader is supposed to be comfortable with Bayesian Networks (BNs): a graphical representation of a joint probability distribution (Conditional Probability Table - CPT) over a set of random variables. BNs have been used in both model-driven and model-based approaches applied to system dependability[5], risk prediction[6] and critical infrastructure protection[7].

2. Background and Related works

2.1. UML Profile and MARTE-DAM

The Unified Modeling Language (UML) is a well known general purpose standardised modelling language for software system specification. UML is equipped with a *profiling* mechanism that allows to customise UML for a particular domain or platform. The UML profiling is actually a *lightweight* meta-modelling technique to extend UML: in such a way, the standard semantics of UML model elements could be refined in a strictly additive manner. Stereotypes, tags and Object Constrain Language (OCL) constraints are the extension mechanisms used to define a UML profile. In particular, a stereotype extends one or more UML meta-classes and can be applied to those UML model elements that are instantiations of the extended meta-classes.

The *UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE)* is an OMG-standard profile that customises UML for the modelling and analysis of non-functional properties (NFP) of real-time embedded systems, such as timing or performance-related properties. MARTE provides a general analysis framework called General Quantitative Analysis Model (GQAM) sub-profile. A key feature of MARTE is the NFP framework used to define new NFP data-types that are necessary to the definition of a specific analysis domain sub-profile.

The *Dependability Analysis and Modeling (DAM)*[8] profile is a specialisation of MARTE-GQAM to enable dependability analysis. DAM is conceived by following a systematic approach, which consists of two main steps: firstly, a dependability *domain model* is built; later, the model is mapped onto UML extensions (i.e., stereotypes, tags and OCL

constraints). Table 1 shows an excerpt of the DAM stereotypes, focusing on those that will be used in the rest of this paper. MARTE-DAM has also been extended towards the maintainability and proved its capability to apply in effective model-driven processes[9,10].

Table 1. DAM stereotypes and tags.

Stereotype	*Inherits / Extends*	*Tags: type*	*Meanings*
System Core			
daService	MARTE::GQAM::gaScenario / -	ssAvail: NFP_Percentage[*] usedResource: Resource[*]	Steady-state availability of the service Resources used by the service (inherited from MARTE)
daComponent	MARTE::GQAM::gaResource / -	ssAvail: NFP_Percentage[*]	Steady-state availability of the component
System Redundancy			
daRedundantStructure	- / UML::Package	FTLevel: NFP_Integer[*]	Level of active replicas needed by a redundant structure to tolerate faults
daSpare	daComponent / -	multiplicity: NFP_Integer substitutesFor: DaComponent[*]	Number of the spare replicas of a component Components the spare part can substitute

2.2. Related works

The problem of defining a proper and effective allocation of VMs onto Physical Machines (PMs) has already been studied in the literature and several results are available[11,12,13], also in the context of funded/open source research projects as mOSAIC [1] and Eucalyptus [2].

Some works use energy to guide the allocation of VMs by pursuing energy efficiency[14,15]. The work of Dougherty et al., while oriented to energy efficiency, also describes a model-driven approach but it does not use standard modelling languages such as UML[16]. The dependability of services provided in the cloud has received recent attention by the scientific community: Huang et al. propose a method based on reliability evaluation by means of series-parallel composition[17] while Yanagisawa et al. deal with this problem by defining a mixed integer programming model[18]. Frincu and Craciun propose a multi-objective optimisation approach for high-availability and cost effective VMs allocation[19].

The work of Zambon et al. presents many similarities to the approach here proposed[20]; the main differences are: (1) the approach proposed here tries to overcome the limitations of the formalisms used in[20] (i.e., Fault Trees and Reliability Block Diagrams); (2) the perspective of the approach here proposed is not limited to planning activities but also to monitoring and enforcement actions; (3) the proposed approach is user-oriented: model-driven principles and techniques are chosen to raise the level of abstraction and to improve the participation of the user in all phases of the cloud service life-cycle, rather than the greatest part of existing scientific works which focused on traditional straight-forward software development[21].

Furthermore, two other works present some interesting features applicable to this paper. De Florio introduces some fault tolerance patterns (e.g., redoing, diversity and redundancy) for software based systems that can be used as basic mechanisms to explore the design space of possible allocations of cloud serves to VMs[22]. From the same work, the concept of distance-to-failure (*dtof*) can be improved in this paper by calculating it on the probability of failure (i.e., the residual probability of having a failure considered the observed faults). Another work is applicable to this paper; it focuses on mechanisms to guarantee software reliability through data integrity both at the design-time and during all the run-time phases of a software system[23].

3. Planning & Monitoring Cloud Resources

Fig. 1 depicts the approach at a glance. The process starts with the EU and a team of software engineers: all of them are involved in the definition of a *Negotiation Model* which represents the set of cloud services requested by the EU and a proper set of VMs providing such services. The UML model is annotated with MARTE-DAM profile in

[1] http://www.mosaic-cloud.eu/
[2] https://www.eucalyptus.com/

order to highlight the availability requirements on requested services. At this point, a model transformation generates an *Optimisation Model* (i.e., a Bayesian Network).

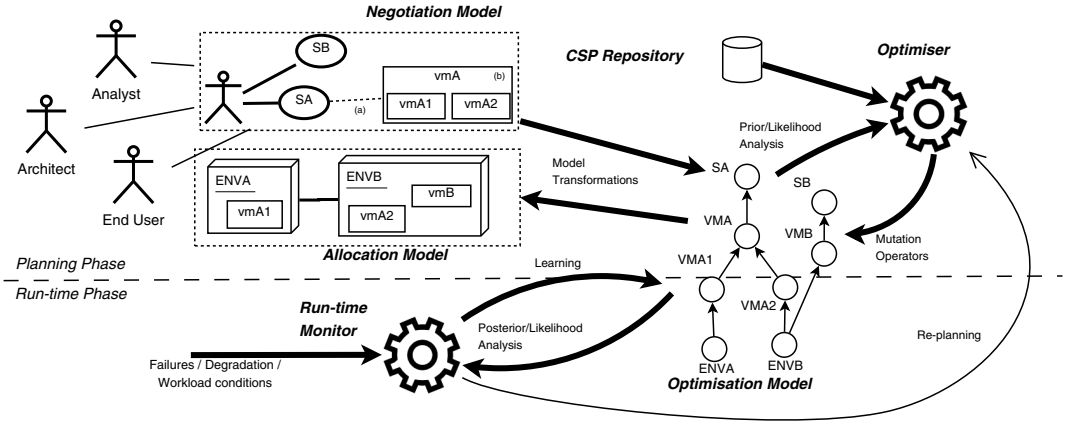

Fig. 1. The availability-oriented planning and monitoring process

The *Optimiser* is in charge of finding one of the best configurations for the infrastructure to provide in terms of: number of replicas for each VMs, which external CSPs to use, allocation of each VMs to the different providers, providing mode of VMs by the different CSPs (in terms of agreed SLAs). To accomplish this task, the Optimiser manipulates the BN Optimisation Model by altering its structure and parameters and evaluating availability indices of the current configurations (via some Mutation Operators): evolutionary algorithms may be exploited such as Genetic Algorithms, Particle Swarm Optimisation, Simulated Annealing, etc. Furthermore, the Optimiser uses a database (the *CSP Repository*) to take the information about known CPSs, the SLAs they offer and the related costs. When the optimal (or a sub-optimal) solution has been found, another model transformation is in charge of generating a high level *Allocation Model* that functionally and non-functionally fulfils the EU requirements. Starting from the Allocation Model, an SLA may be written and used for the final agreement with EU.

The described process is just a part of a wider approach aiming at supporting not only the design phase by providing a first allocation of cloud resources; furthermore, the defined models may be then used to support run-time phase by supervising the operation of the cloud service. The approach takes into consideration a *run-time monitor* which collects availability-related events and uses them in conjunction with the Optimisation model which plays the role of event correlator. Updated information about the residual probability of having a failure may be inferred by analysing the Optimisation Model in a proper way: when such probability reaches a pre-violation threshold and the current allocation is judged to be no more sustainable with respect to the signed SLA, a more robust allocation of cloud services on VMs may be required to the Optimiser. According to the granularity of the model, this phase can consider failures of some nodes as well as information about the workload and the performance degradation of single sub-services.

To accomplish these tasks, three different ways of analysing BNs are used: *Prior probability*, that is the way of computing the steady state availability of the cloud services in absence of faulty events or anomalies; *Posterior probability*, that is the way of computing the steady state availability when a faulty event or an anomaly occurs; *Likelihood*, that computes the expected fault probability of single VM and software components replicas under the hypothesis of a service failure. Hence, Prior probability and Likelihood are used during the *Planning Phase* by evaluating the steady state availability of the candidate solution (Prior probability) and by finding the most likable cause to a possible failure of the system in order to produce new candidate solutions (Likelihood). At *Run-time Phase*, the same uses are done of Posterior probability and Likelihood with the difference that real events are available and then some nodes of the BN model can be "observed".

Another important feature of the BNs may be used during the *Run-time Phase*. Scientific literature produced many works where BNs are at the centre of learning algorithms that receive as input data and give back in output both a BN model with adjusted parameters (i.e., the correlation weights in CPTs) or a BN model modified in its structure[24].

For a sake of clarity, we present here a little example which will be used in the rest of the paper. Let us consider the case of a cloud based web service composed by an application (S1) and a management service. The application, a simple on-line game, is provided by a classical three tiers application (a DMBS (C1), an application server (C2) and a web server (C3)). The management service simply provides statistics (S2) and an administration application (S3) to the site administrator. Both the services use the DBMS as well as two dedicated components (respectively C4 and C5); S2 also uses the web server. Let us suppose for simplicity that there are three VMs: one encapsulating the DBMS (VM1), one for the web server (VM2) and one collecting all the application servers (VM3). While it is tolerable a low availability level of the analytical and statistical service ($A_{S2} = 0.9$), both management service and the game itself must present higher availability levels (respectively $A_{S3} = 0.99$ and $A_{S1} = 0.995$). Let us moreover suppose that the availability of software components is 0.999 and the availability of VMs is 0.99.

4. The Model Levels

4.1. The Negotiation Model

The Negotiation Model counts two main diagrams: a *Use Case diagram* (UCD) and a *Component diagram* (CD). For each service to negotiate and to put in the cloud environment, a Use Case (UC) is created. The Use Case diagram, as prescribed by the UML language, may contain standard relationships between use cases: `<<include>>` and `<<extend>>`. Then, a Component diagram must be created, in order to describe the structure of the software components which are in charge of providing services described in the Use Case diagram. Each component may be a software component in the traditional meaning or may be a Virtual Machine.

Once the UML model is defined, it can be annotated by using MARTE-DAM stereotypes and related tagged values: in this way, the modeller can capture the availability requirements as well as dependencies between software components and provided services. The annotation of the model mainly uses the stereotypes of Table 1: `<<daService>>` is used to annotate an offered service; `<<daComponent>>` is used to annotate the software components; and `<<subsystem>>` (a UML standard profile L2 stereotype) is used to annotate VMs. A possible Negotiation Model of the example described in Section 3 is depicted in Fig. 2.

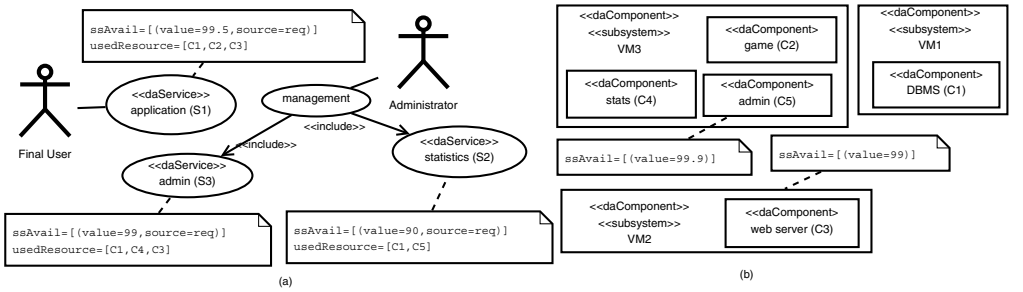

Fig. 2. Negotiation Model of the running example: (a) Use Case Diagram (b) Component Diagram

4.2. The Optimisation Model

This paper does not formally describe the Model Transformations: notwithstanding, the mapping between the used UML constructs and BN elements is sketched. Firstly, a starting BN model is generated by applying these rules: **[FR1]** for each UC annotated as `<<daService>>`, a BN node is generated. When a UC includes/extends another UC, an arc in drawn from the BN node translating the included/extended UC to the one related to the including/extending UC; **[FR2]** for each Component annotated as `<<daComponent>>`, a BN node is generated. When a Component is included into another one, an arc is drawn from the BN node translating the included Component to the one related to the including Component; **[FR3]** an arc is drawn from a BN node obtained from C applying *FR2* to a BN node obtained from S applying *FR1* when the *usedResource* tagged value of the S includes C.

All the BN nodes are binary: they may be *up* or *down*. For each BN node, a proper CPT is generated according to the meaning of the node and the relationships with its parents. An example is constituted by a generic `<<daService>>` S which includes $S_1, ..., S_N$ other services and uses $C_1, ..., C_M$ `<<daComponent>>`s. In this case, S is available (*up*) if all the included services and used components are *up*.

Once the model is built, the Optimiser is in charge of defining the best model structure and parameters able to maximise the availability of the services and to contain the costs of the different considered deployments. The Optimiser may work according to the generic evolutionary algorithm where new tentative solutions are created on the basis of the precedent ones, by applying some mutation operators and evaluating the fitness of the new solutions. The information on how to deploy VMs are contained in a database, the CSP Repository: in such repository there is a list of all the external CSPs available to host VMs, the cost of the deploy and the related availability CSP is able to accomplish. Moreover, physical (also in-house) machines may be added to this repository.

In the context of the optimisation process, the role of the BN model is to evaluate the fitness of the solution by calculating the steady state availability of the services (i.e., the prior probability that the nodes representing the services are *up*). Furthermore, by using likelihood analysis the most plausible causes to a supposed *down* state of the services are found; availability bottlenecks of the systems are detected and used as main directions to improve the quality of the candidate solution. In other words, by calculating the probability that each node (other services, software components, VMs or PMs) has in causing the service level failure, the Optimiser may improve the existing solution by focusing on the most likable causes.

The set of mutation operators used by the optimisation engine in the generation of new solutions may include the following: **[OR1]** (*insertion*): this operator extracts a CSP from the CSP Repository and allocate a VM to this CSP. The effects of this action on the BN model are the creation of a new BN node (representing the CSP) and of a new arc from this node to the one related to the VM; **[OR2]** (*allocation*): this operator is a simplified version of the previous one. It links an unallocated VM to an existing CSP; **[OR3]** (*migration*): it allocates a yet allocated VM to another CSP; **[OR4]** (*redundancy*): to increase the availability of some components, redundancy may be necessary. This operator creates replicas of software artifacts (components and VMs). In relation to the BN model, this operator would duplicate the subnet related to the replicated artifact, for all the artifacts contained in it.

Fig. 3(a) shows the structure of the BN model obtained by translating the Negotiation Model in Fig. 2 while Fig. 3(b) depicts the structure of a possible BN model obtained by optimising the first model presenting the following features. First, three external CSPs are present, the third of them hosting two VMs. Second, *VM1* is redundant (and so also *C1*) where replicas of *VM1* (respectively *C1*) are *VM1a* and *VM1b*. Finally, the node *C1* has *C1a* and *C1b* as parents and it has a CPT implementing a logical OR (i.e., the node is *up* when at least one replica is *up*).

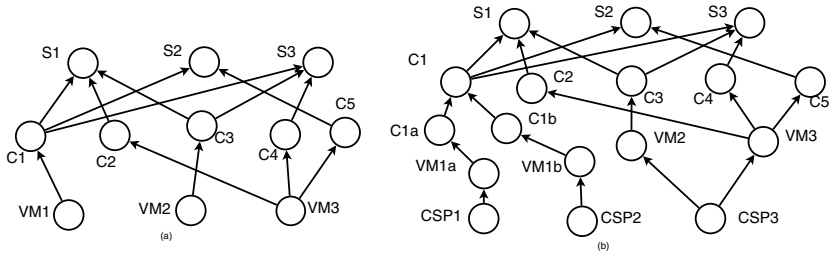

Fig. 3. Optimisation Model of the running example: (a) initial (b) possible solution

4.3. The Allocation Model

Once the Optimiser has found a possible VMs deployment, a Model Transformation may generate an annotated UML model able to capture the deployment relationships among software components, VMs and external CSPs: the Allocation Model. The Allocation Model is constituted by two UML diagrams: a Component and a Use Case diagrams, refining those contained in the Negotiation Model, and a Deployment Diagram (DD) where each Virtual Machine of the CD is contained in UML Node representing the external CSP or the physical machine hosting the VM.

Hence, this is a refining transformation that has two inputs (the Negotiation Model and the Optimisation Model) and an output (the Allocation Model) where some UML model elements are added/modified. More in details, the following rules constitute the logic of the transformation process: **[RR1]** for each application of the *OR4* rule, let $C^R = C^R_1, \ldots, C^R_k$ be the set of the components that have been redundant and C^R_i the most external of such components according to the subcomponent relationship. The Component Diagram is modified by (a) creating a new UML Component, (b) inserting this component as well as the original one into a newly created UML Package. The new component is annotated using the `<<daSpare>>` MARTE-DAM stereotype setting the *multiplicity* tagged value to the overall number of replicas minus one. The package is annotated with the `<<daRedundantStructure>>` stereotype (see Table 1); **[RR2]** for each `<<daService>>` of the UCD, the *ssAvail* tagged value is improved by reporting the actual value of the service availability as calculated by solving the BN model; **[RR3]** for each BN node generated by *OR1*: (a) a UML Node N is created and annotated, (b) C_i of the Component Diagram is deployed by N if the BN node generating N is a parent of the one generated by C_i. Moreover, N is annotated with three stereotypes: UML's `<<device>>`, MARTE-DAM's `<<daComponent>>` in order to capture the availability information of the deployment and MARTE's `<<HW_Component>>` in order to model cost information (by means of the *price* tagged value).

The Allocation Model related to the example is depicted in the Fig. 4.

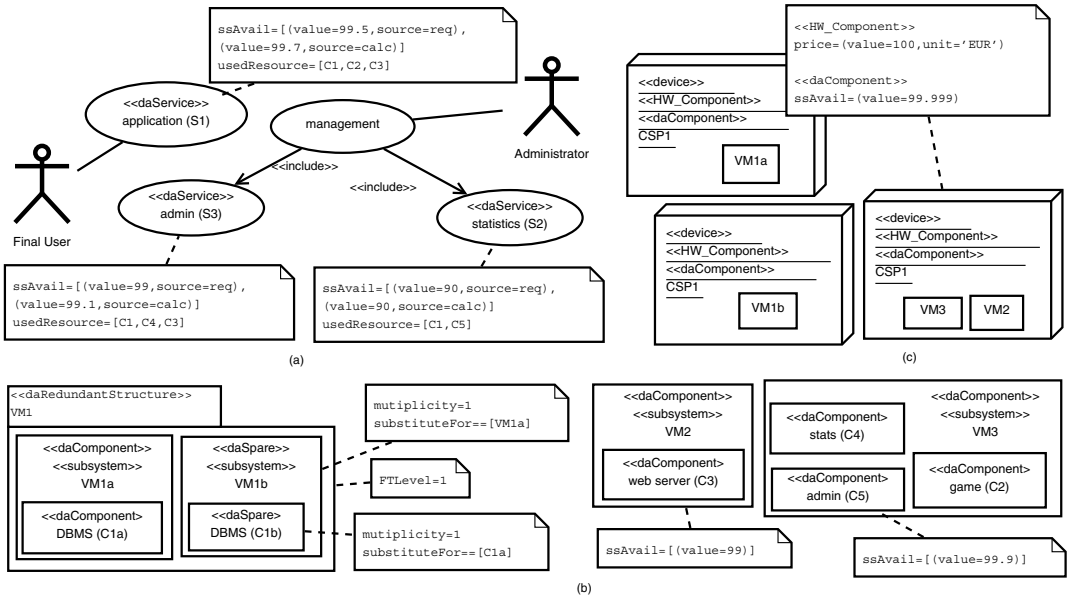

Fig. 4. Allocation Model of the running example: (a) Use Case Diagram, (b) Component Diagram (c) Deployment Diagram

5. Discussion and Future Developments

The paper proposes a model-driven approach for the automatic negotiation and resource allocation for availability critical cloud services. The modelling approach uses both profiled UML models as well as Bayesian Networks: this last formalism is used not only to evaluate the availability of the configurations but also to address the exploration of the design space in a more efficient way. Despite the present paper focuses on the usage of UML and BNs, the approach also works when implementing the Negotiation, the Optimisation and the Allocation models with other languages.

The paper is a first step of a research work whose objective is to create a support system also for the operational phases of the cloud application life-cycle. A wider automatic process is sketched that, starting from the allocated cloud resources, defines the scope of a monitoring system able to capture real world events and to use the Optimisation Model by setting observations and inferring the residual probability of failure. In the case, that such "distance to

failure" results under a defined threshold, a reconfiguration of the cloud service allocation may be requested by this module. The application of learning algorithms would improve the ability of the system to react to faults and to produce more robust allocation solutions as the cloud service and the monitoring system run.

Future research effort will be oriented to understand how these models can be used to improve the service availability during the normal functioning of the system. To achieve this step, learning techniques already available for both BNs and their extension, Dynamic BNs, will be experimented. Moreover, other dependability and security features will be added in the modelling and transformational process to create more comprehensive BN models taking into account different non-functional features.

References

1. Dekker, M., Hogben, G.. Survey and analysis of security parameters in cloud slas across the european public sector. Tech. Rep.; European Network and Information Security Agency; 2011.
2. Kritikos, K., Pernici, B., Plebani, P., Cappiello, C., Comuzzi, M., Benrernou, S., et al. A survey on service quality description. *ACM Comput Surv* 2013;**46**(1):1:1–1:58. URL: http://doi.acm.org/10.1145/2522968.2522969. doi:10.1145/2522968.2522969.
3. Schmidt, D.C.. Guest editor's introduction: Model-driven engineering. *Computer* 2006;**39**(2):25–31. doi:10.1109/MC.2006.58.
4. Rak, M., Suri, N., Luna, J., Petcu, D., Casola, V., Villano, U.. Security as a service using an SLA-based approach via SPECS. In: *Cloud Computing Technology and Science (CloudCom), IEEE 5th Int. Conf. on*; vol. 2. 2013, p. 1–6. doi:10.1109/CloudCom.2013.165.
5. Langseth, H., Portinale, L.. Bayesian networks in reliability. *Reliability Engineering and System Safety* 2007;**92**(1):92–108. URL: http://www.scopus.com/inward/record.url?eid=2-s2.0-33748310590&partnerID=40&md5=89439322d7a37ce69a6901e7cee65dcf. doi:10.1016/j.ress.2005.11.037; cited By 170.
6. Weber, P., Medina-Oliva, G., Simon, C., Iung, B.. Overview on bayesian networks applications for dependability, risk analysis and maintenance areas. *Eng Appl Artif Intell* 2012;**25**(4):671–682. doi:10.1016/j.engappai.2010.06.002.
7. Marrone, S., Nardone, R., Tedesco, A., D'Amore, P., Vittorini, V., Setola, R., et al. Vulnerability modeling and analysis for critical infrastructure protection applications. *International Journal of Critical Infrastructure Protection* 2013;**6**(34):217–227.
8. Bernardi, S., Merseguer, J., Petriu, D.C.. A dependability profile within MARTE. *Software and System Modeling* 2011;**10**(3):313–336.
9. Bernardi, S., Flammini, F., Marrone, S., Merseguer, J., Papa, C., Vittorini, V.. Model-Driven Availability Evaluation of Railway Control Systems. In: *Computer Safety, Reliability, and Security*; vol. 6894 of *LNCS*. Springer. ISBN 978-3-642-24269-4; 2011, p. 15–28.
10. Bernardi, S., Flammini, F., Marrone, S., Mazzocca, N., Merseguer, J., Nardone, R., et al. Enabling the usage of UML in the verification of railway systems: The DAM-rail approach. *Reliability Engineering & System Safety* 2013;**120**(0):112 – 126.
11. Calheiros, R., Ranjan, R., Beloglazov, A., De Rose, C., Buyya, R.. Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software - Practice and Experience* 2011;**41**(1):23–50.
12. Xiao, Z., Song, W., Chen, Q.. Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Transactions on Parallel and Distributed Systems* 2013;**24**(6):1107–1117.
13. Ezugwu Absalom E., S.M.B., Junaidu, S.B.. Virtual machine allocation in cloud computing environment. *International Journal of Cloud Applications and Computing (IJCAC)* 2013;**3**(2):47–60.
14. Beloglazov, A., Abawajy, J., Buyya, R.. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems* 2012;**28**(5):755–768.
15. Quang-Hung, N., Thoai, N., Son, N.. EPOBF: Energy efficient allocation of virtual machines in high performance computing cloud. In: *Transactions on Large-Scale Data and Knowledge-Centered Systems XVI*; LNCS. ISBN 978-3-662-45946-1; 2014, p. 71–86.
16. Dougherty, B., White, J., Schmidt, D.. Model-driven auto-scaling of green cloud computing infrastructure. *Future Generation Computer Systems* 2012;**28**(2):371–378.
17. Huang, C., Chen, J., Zhang, L., Zhu, Q.. Architecting dependable virtual computing system with considering error propagation. *Journal of Computational Information Systems* 2013;**9**(4):1593–1601.
18. Yanagisawa, H., Osogami, T., Raymond, R.. Dependable virtual machine allocation. In: *INFOCOM, 2013 Proceedings IEEE*. 2013, p. 629–637. doi:10.1109/INFCOM.2013.6566848.
19. Frincu, M.E., Craciun, C.. Multi-objective meta-heuristics for scheduling applications with high availability requirements and cost constraints in multi-cloud environments. In: *Utility and Cloud Computing (UCC), 2011 4th IEEE Int. Conf. on*. IEEE; 2011, p. 267–274.
20. Zambon, E., Etalle, S., Wieringa, R.J.. A2thos: Availability analysis and optimisation in slas. *Int J Netw Manag* 2012;**22**(2):104–130. URL: http://dx.doi.org/10.1002/nem.790. doi:10.1002/nem.790.
21. Ardagna, D., di Nitto, E., Mohagheghi, P., Mosser, S., Ballagny, C., D'Andria, F., et al. Modaclouds: A model-driven approach for the design and execution of applications on multiple clouds. In: *Modeling in Software Engineering (MISE), 2012 ICSE Workshop on*. 2012, p. 50–56. doi:10.1109/MISE.2012.6226014.
22. De Florio, V.. Software assumptions failure tolerance: Role, strategies, and visions. In: *Architecting Dependable Systems VII*; vol. 6420 of *LNCS*. ISBN 978-3-642-17244-1; 2010, p. 249–272. doi:10.1007/978-3-642-17245-8_11.
23. De Florio, V., Blondia, C.. On the requirements of new software development. *Int J Bus Intell Data Min* 2008;**3**(3):330–349. URL: http://dx.doi.org/10.1504/IJBIDM.2008.022138. doi:10.1504/IJBIDM.2008.022138.
24. Neapolitan, R.. *Learning Bayesian Networks*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.; 2003. ISBN 0130125342.